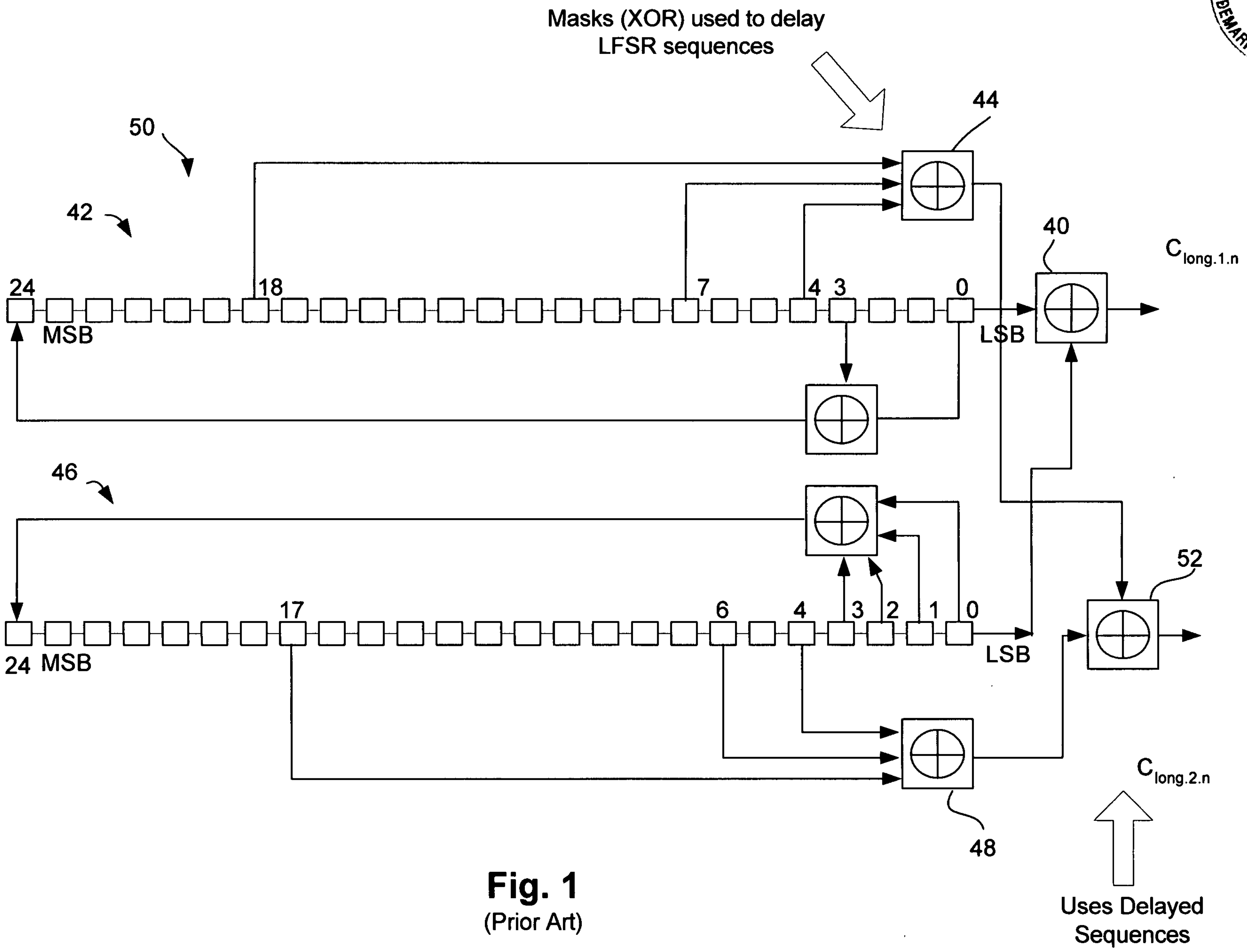

Fig. 1
(Prior Art)

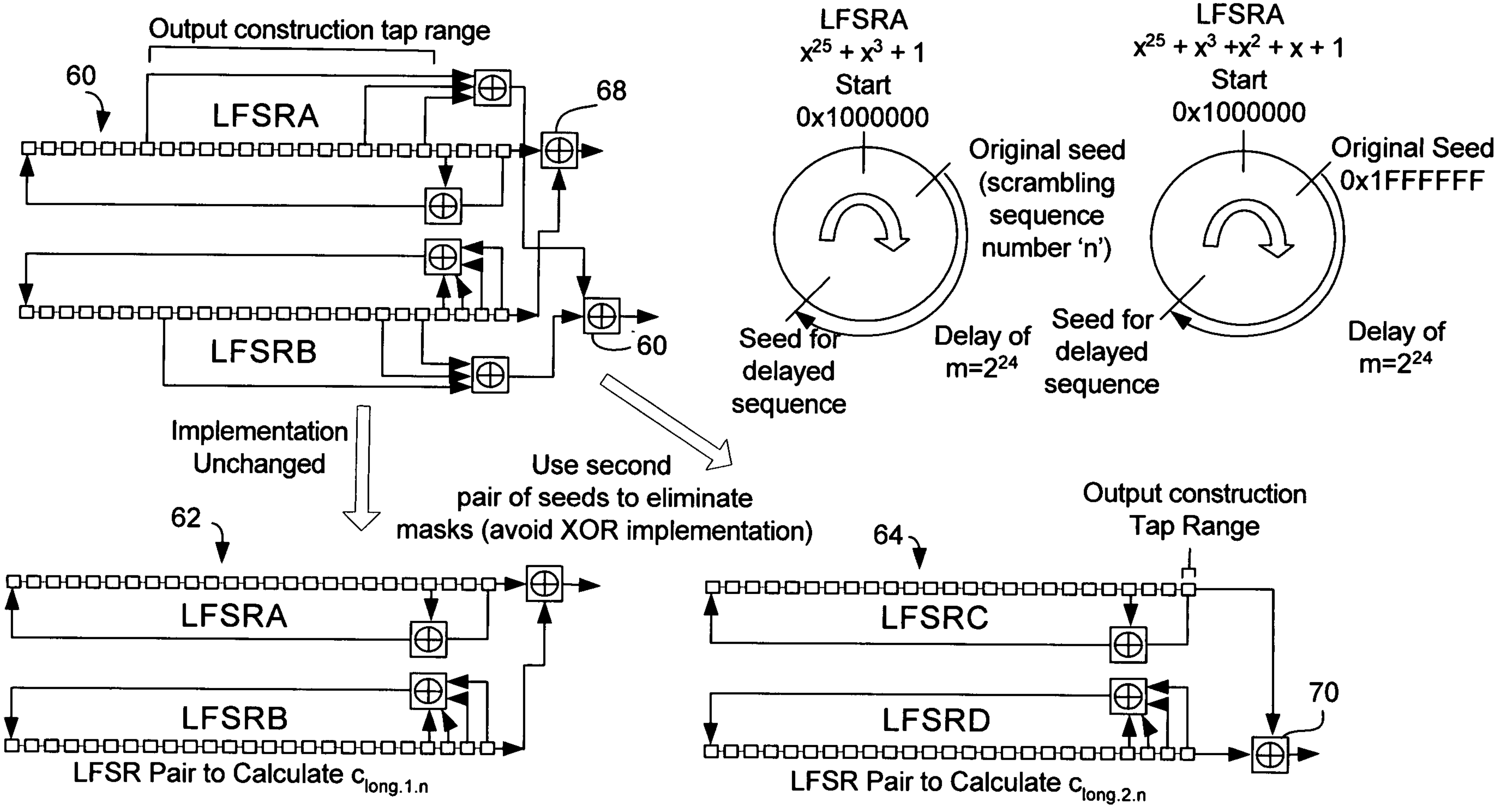
Output construction tap range
60
LFSRA
68
LFSRB
60
Implementation Unchanged
Use second pair of seeds to eliminate masks (avoid XOR implementation)
LFSRA
$x^{25} + x^3 + 1$
Start
0x1000000
Original seed (scrambling sequence number 'n')
Seed for delayed sequence
Delay of $m=2^{24}$
LFSRA
$x^{25} + x^3 + x^2 + x + 1$
Start
0x1000000
Original Seed 0x1FFFFFF
Seed for delayed sequence
Delay of $m=2^{24}$
62
LFSRA
LFSRB
LFSR Pair to Calculate $c_{long.1.n}$
64
Output construction Tap Range
LFSRC
LFSRD
70
LFSR Pair to Calculate $c_{long.2.n}$

Fig. 2

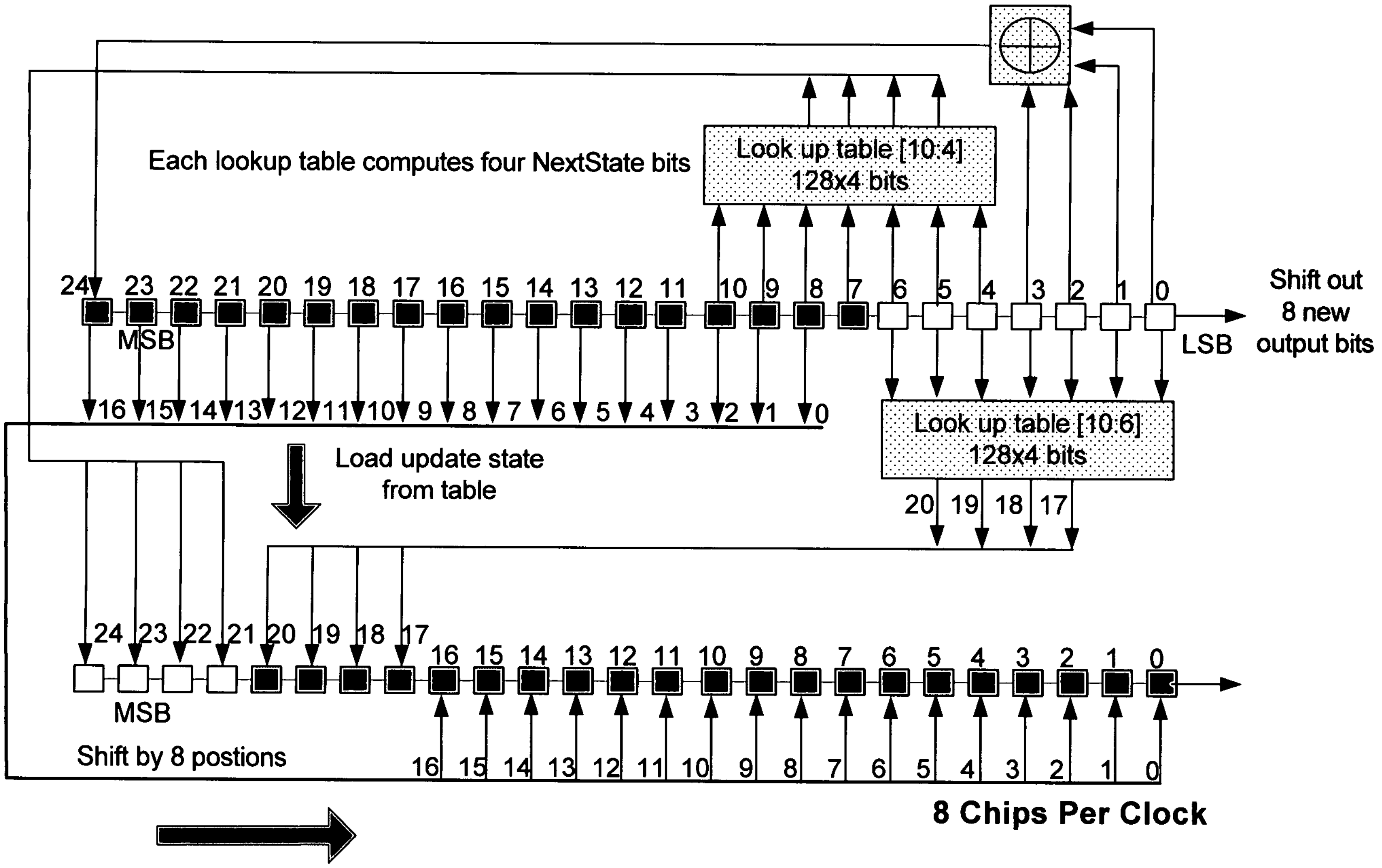
Each lookup table computes four NextState bits
Look up table [10:4]
128x4 bits
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
MSB
LSB
Shift out
8 new
output bits
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Look up table [10:6]
128x4 bits
20
19
18
17
Load update state
from table
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
MSB
Shift by 8 postions
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
8 Chips Per Clock

Fig. 3

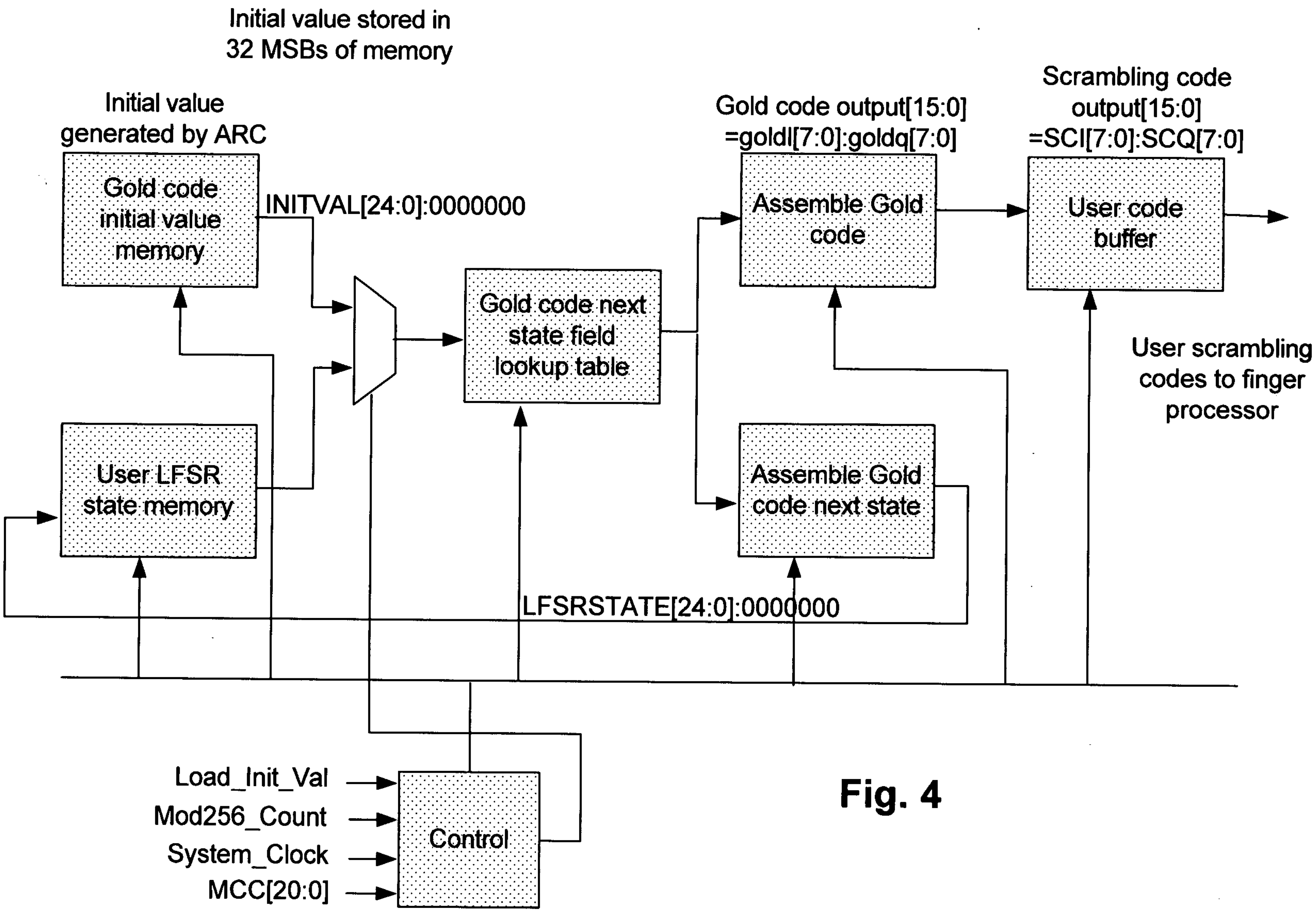
Initial value stored in
32 MSBs of memory
Initial value
generated by ARC
Gold code initial value memory
INITVAL[24:0]:0000000
User LFSR state memory
Gold code next state field lookup table
Gold code output[15:0]
=goldI[7:0]:goldq[7:0]
Assemble Gold code
Assemble Gold code next state
Scrambling code
output[15:0]
=SCI[7:0]:SCQ[7:0]
User code buffer
User scrambling codes to finger processor
LFSRSTATE[24:0]:0000000
Load_Init_Val
Mod256_Count
System_Clock
MCC[20:0]
Control

Fig. 4

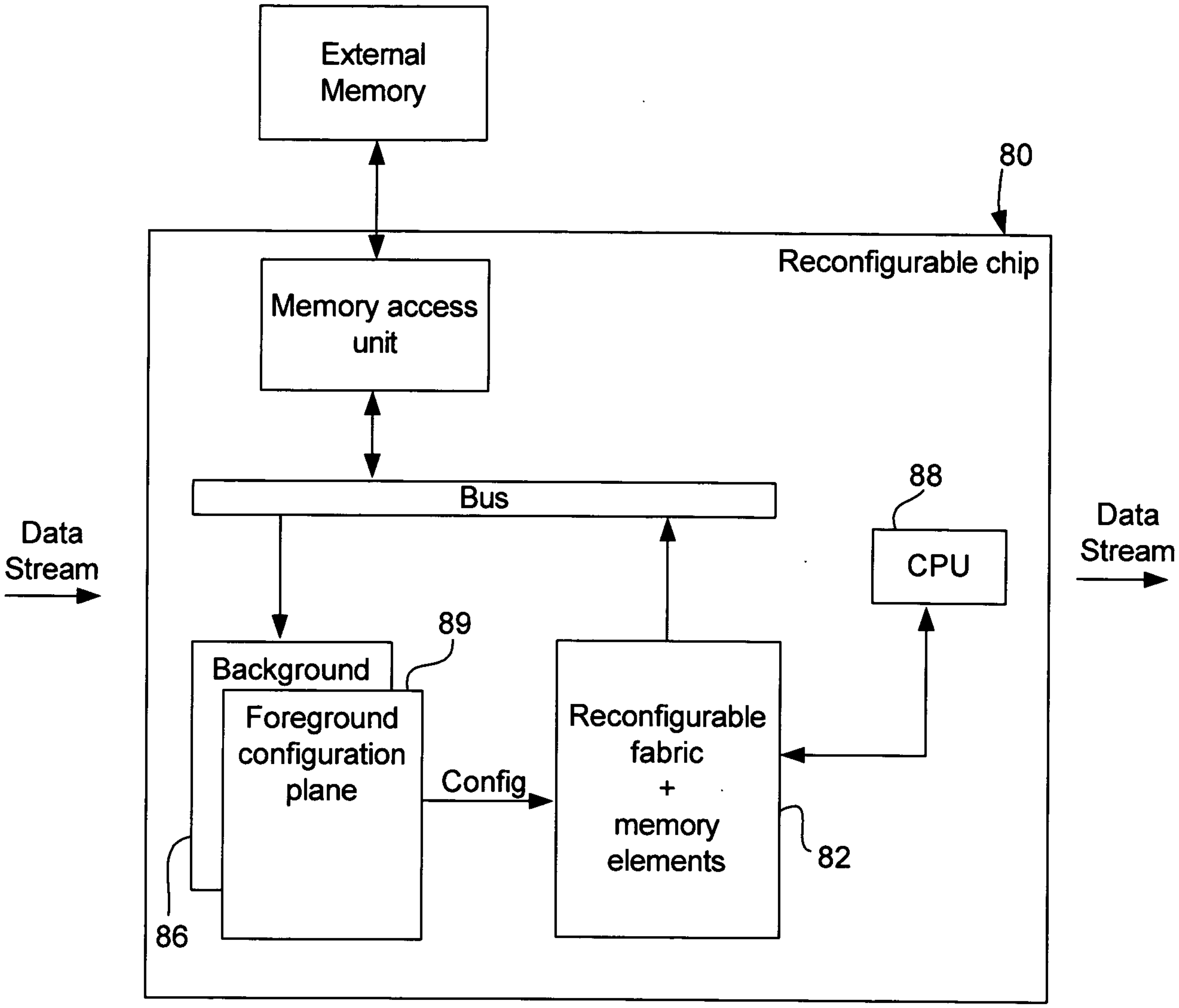
External Memory
80
Reconfigurable chip
Memory access unit
Bus
88
CPU
Data Stream
Data Stream
89
Background
Foreground configuration plane
Config
Reconfigurable fabric + memory elements
82
86

Fig. 5

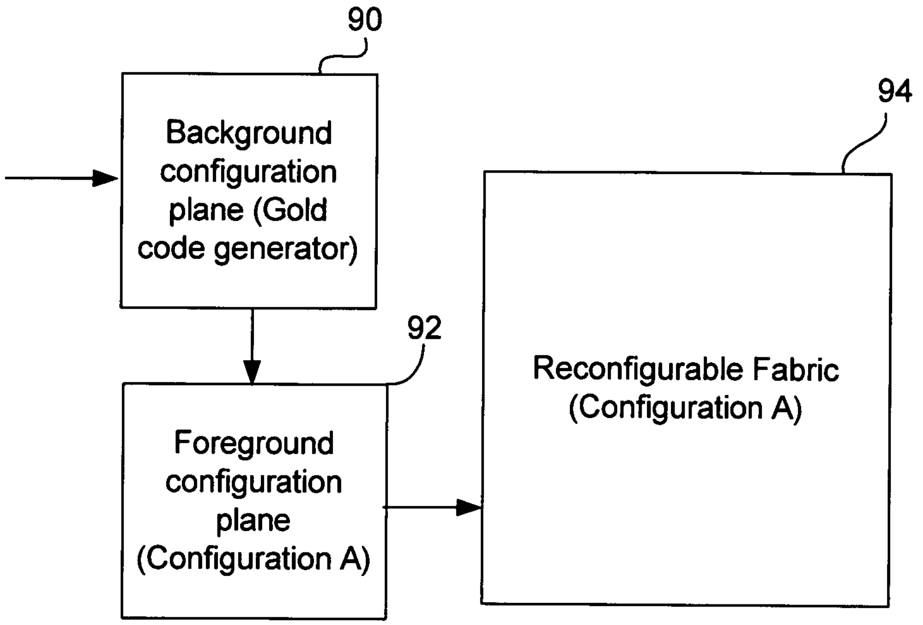
90
94
Background configuration plane (Gold code generator)
92
Reconfigurable Fabric (Configuration A)
Foreground configuration plane (Configuration A)

Fig. 6A

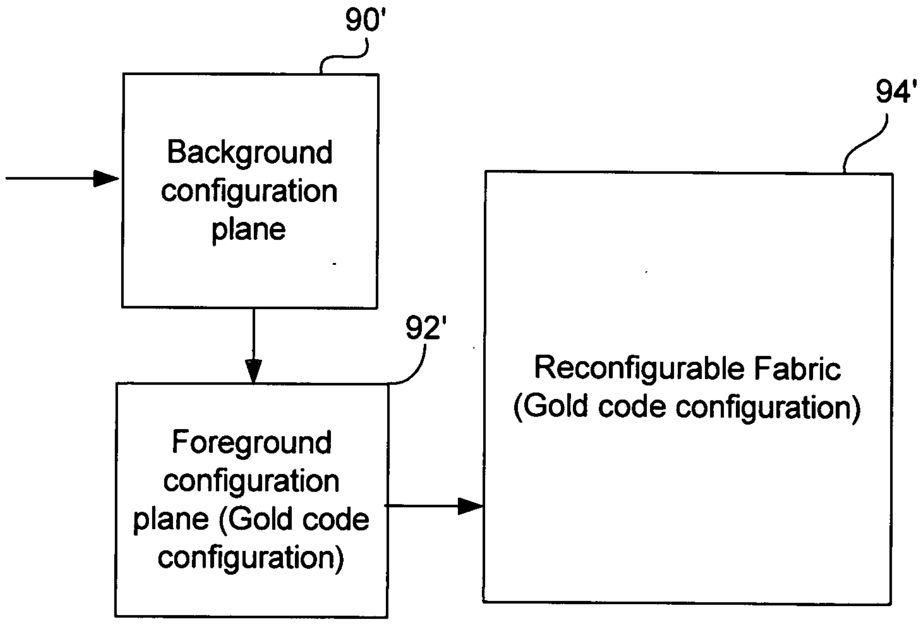
90'
94'
Background configuration plane
92'
Reconfigurable Fabric (Gold code configuration)
Foreground configuration plane (Gold code configuration)

Fig. 6B

$C_{long1.n}$ = LFSRA[7:0] XOR LFSRB[7:0]
Let us define LFSRC'[i] = LFSRC[2[I/2]]

$C_{long.n}(i) = C_{long.n}(i)(1+j(-1)^i(c_{long2,}n(2[i/2])$ (From 3G TS25.213)
Multiplying bits by +1/-1 is the same as XOR for 0s and 1s.
XORing by 0xAA can be used in place of the $(-1)^i$ term.
In binary representation, the Scrambling code $C_{long.n}$ becomes:

$C_{long.n}$[7:0] = $C_{long1.n}$[7:0](1+j(0xAA) XOR $C'_{long2.n}$[7:0])
$C_{long.n}$[7:0] = LSFRA[7:0] XOR LFSRB[7:0]
+J(LFSRA[7:0] XOR LFSRB[7:0] XOR 0xAA XOR LFSRC'[7:0] XOR LFSRD'[7:0])
$C_{long.n}$[7:0] = SCI[7:0] = Jscq[7:0]

Let us define LFSRD"[7:0] = 0xAA XOR LFSRD'[7:0], then:
$C_{long.n}$[7:0] = (LFSRA[7:0] XOR LFSRB[7:0])
+ j(LFSRA[7:0] XOR LFSRB[7:0] XOR LFSRC'[7:0] XOR LFSRD"[7:0])

We use a lookup table to compute LFSRC'[7:0] and LFSRD"[7:0])

Fig. 7

Gold code generator lookup[6:0] definitions

At address 4n+0: OUT[7:0] = Next StateA[3:0]:PASSA[3:0]
OUT[7] = IN[6] XOR IN[3]
OUT[6] = IN[5] XOR IN[2]
OUT[5] = IN[4] XOR IN[1]
OUT[4] = IN[3] XOR IN[0]
OUT[3] = IN[3]
OUT[2] = IN[2]
OUT[1] = IN[1]
OUT[0] = IN[0]
	At address 4n+2: OUT[7:0] = Next StateC[3:0]:LFSRC'[3:0]
	OUT[7] = IN[6] XOR IN[3]
	OUT[6] = IN[5] XOR IN[2]
	OUT[5] = IN[4] XOR IN[1]
	OUT[4] = IN[3] XOR IN[0]
	OUT[3] = IN[3]
	OUT[2] = IN[2]
	OUT[1] = IN[1]
	OUT[0] = IN[0]

At address 4n+1: OUT[7:0] = Next StateB[3:0]:PASSA[3:0]
OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3]
OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2]
OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1]
OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0]
OUT[3] = IN[3]
OUT[2] = IN[2]
OUT[1] = IN[1]
OUT[0] = IN[0]
	At address 4n+3: OUT[7:0] = Next StateD[3:0]:LFSRD"[3:0]
	OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3]
	OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2]
	OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1]
	OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0]
	OUT[3] = /IN[2]
	OUT[2] = IN[2]
	OUT[1] = /IN[0]
	OUT[0] = IN[0]

Fig. 8A

Gold code generator lookup[10:4] definitions

At address 4n+0: OUT[7:0] = IN[7.4] Next StateA[7:4]
OUT[7] = IN[3]
OUT[6] = IN[2]
OUT[5] = IN[1]
OUT[4] = IN[0]
OUT[3] = IN[6] XOR IN[3]
OUT[2] = IN[5] XOR IN[2]
OUT[1] = IN[4] XOR IN[1]
OUT[0] = IN[3] XOR IN[0]
	At address 4n+0: OUT[7:0] = IN[7:4] Next StateA[7:4]
	OUT[3] = IN[2]
	OUT[2] = IN[2]
	OUT[1] = IN[0]
	OUT[0] = IN[0]
	OUT[7] = IN[6] XOR IN[3]
	OUT[6] = IN[5] XOR IN[2]
	OUT[5] = IN[4] XOR IN[1]
	OUT[4] = IN[3] XOR IN[0]

At address 4n+1: OUT[7:0] = IN[7:4] Next StateB[7:4]
OUT[7] = IN[6]
OUT[6] = IN[5]
OUT[5] = IN[4]
OUT[4] = IN[3]
OUT[3] = IN[3] XOR IN[5] XOR IN[4] XOR IN[3]
OUT[2] = IN[2] XOR IN[4] XOR IN[3] XOR IN[2]
OUT[1] = IN[1] XOR IN[3] XOR IN[2] XOR IN[1]
OUT[0] = IN[0] XOR IN[2] XOR IN[1] XOR IN[0]
	At address 4n+1: OUT[7:0] = IN"[7:4]Next StateB[7:4]
	OUT[3] = /IN[2]
	OUT[2] = IN[2]
	OUT[1] = /IN[0]
	OUT[0] = IN[0]
	OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3]
	OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2]
	OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1]
	OUT[4] = IN[] XOR IN[2] XOR IN[1] XOR IN[0]

Fig. 8B